CPE 223 Digital circuit design

Final Project Report: “IA Journey”

Section **B**

Mr. Rachasak	Ragkamnerd	57070501075
Mr. Suttiwat	Songboonkaew	57070501079

Instructor Mr. Surapont Toomnark & Mr. Pipat Supasirisun

ชุดคำสั่งที่ใช้ในโครงงานนี้และรายงานสามารถเข้าถึงได้ที่

 Github https://github.com/itpcc/FPGA-IA-Journy-game/

วัตถุประสงค์

เพื่อสร้างเกมส์ “IA Journey” ซึ่งเป็นเกมให้หลบแมวซึ่งเป็นศัตรูที่กำเนิดอย่างสุ่มทุกๆ 5 วินาที และสามารถยิงปืนเพื่อลบแมวออกได้ตามจังหวะที่กำหนด นับคะแนนทุกๆ 2.4 วินาที เพิ่ม 1 คะแนน โดยส่งข้อมูลการเคลื่อนที่และการสั่งยิงปืนผ่านปุ่มกดและแสดงผลบน VGA-mode color display

อุปกรณ์

- โปรแกรม Xilinx ISE 14.7 software
- APEX Discovery III (Spartan 3 XC3S400-4TQ144C) FPGA board
- Push-button
- VGA port module

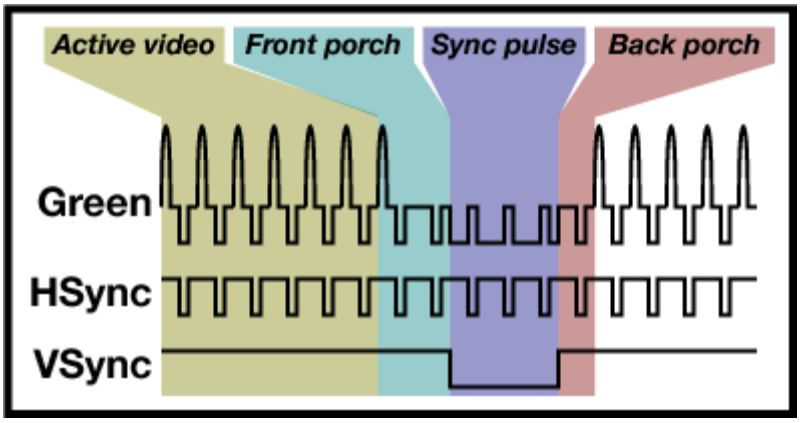
ทฤษฎีที่เกี่ยวข้อง

การแสดงผลภาพแบบ VGA

VGA คือมาตรฐานการแสดงผลภาพบนหน้าจอสีโดย IBM แรกเริ่มเพื่อใช้ในคอมพิวเตอร์ IBM PS/2 โดยการส่งสัญญาณที่สำคัญ 5 สัญญาณ (Hinner, 2007) ได้แก่:

- Vertical Sync เป็นสัญญาณดิจิตอลเพื่อกำหนด Frame rate และกำหนดให้ Electron beam ในจอแบบ CRT เริ่มกวาดภาพที่ด้านบนจอ
- Horizontal Sync เป็นสัญญาณดิจิตอลเพื่อกำหนดจังหวะให้ Electron beam ในจอแบบ CRT เริ่มกวาดภาพที่ด้านซ้ายของจอในแถวถัดไป
- สัญญาณสี R, G, B ในรูปแบบสัญญาณ Analog 0-0.7 โวลต์ ยิ่งโวลต์สูง สีนั้นยิ่งสว่าง

โดยทั่วไปการส่งสัญญาณภาพมักมีช่วงรอยต่อท้ายเฟรมปัจจุบัน หรือ “Front porch” และก่อนหน้าเฟรมถัดไป หรือ “Back porch” เพื่อป้องกันภาพเหลื่อมกันและทำให้แรงดันไฟฟ้ากลับมาเสถียรในกรณีจอภาพเก่า (Pemberton, 2014) และปกติในช่องสัญญาณสีจะรับ input ที่ 75 Ω หากต้องจ่ายสัญญาณไฟจากแหล่งกำเนิด 3.3 โวลต์ ต้องต่อตัวต้านทานขนาด 270 Ω อนุกรมด้วยเพื่อสร้างตัวแบ่งแรงดันไฟฟ้าให้ปลายทางได้แรงดันไฟฟ้าที่ไม่เกิน 0.7 โวลต์ (Nicolle, 2014)



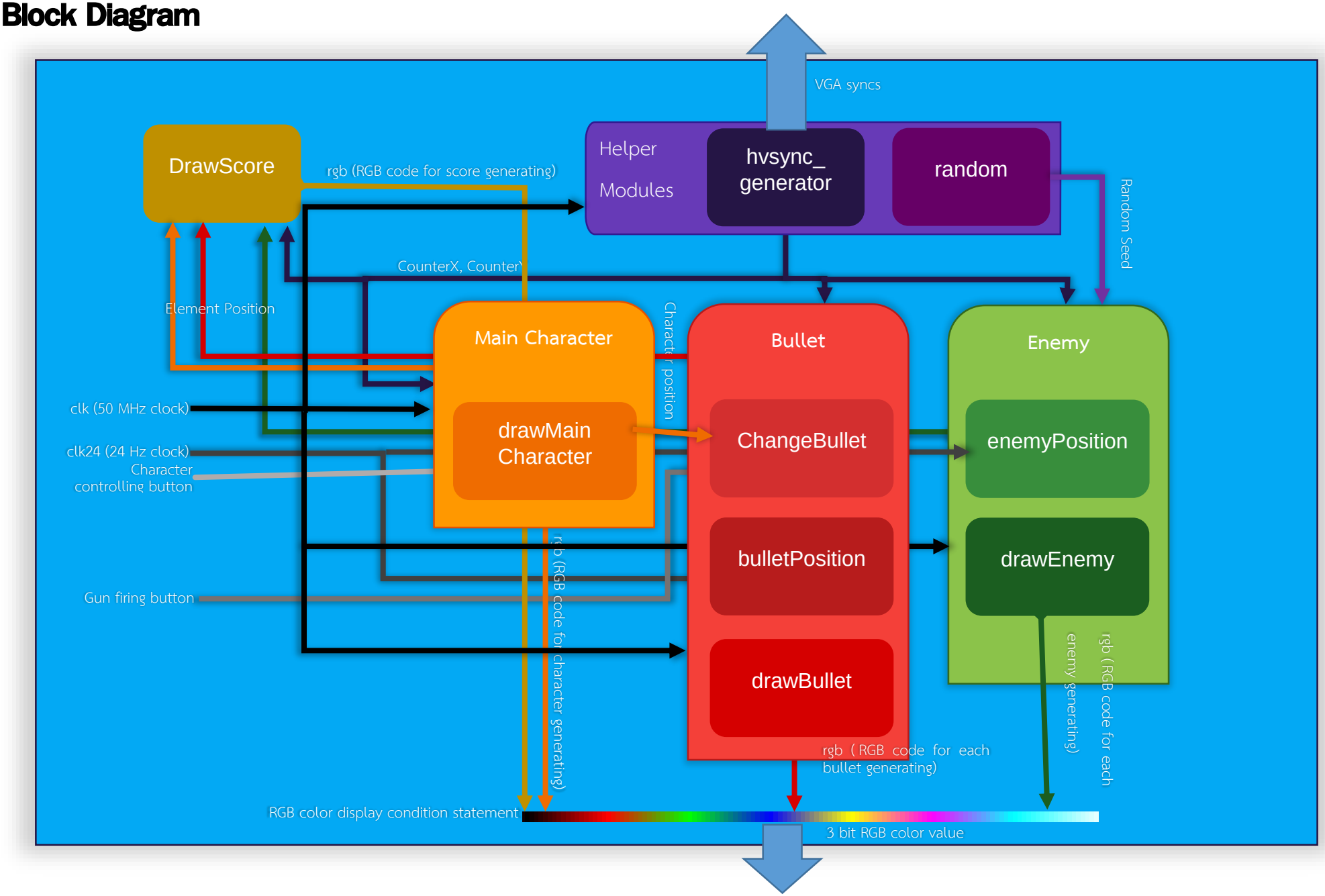
รูปที่ 1 แสดงตัวอย่าง Timing diagram ของสัญญาณภาพแบบ VGA

สำหรับขนาดภาพ VGA มาตรฐาน*อุตสาหกรรม*มีค่าดังนี้ (Hinner, 2007)

- ขนาดภาพ 640x480 pixel 60 เฟรม/วินาที
- ความถี่สัญญาณนาฬิกา 25.175 MHz
- ความถี่สัญญาณ Horizontal sync 31469 Hz
- ความถี่สัญญาณ Vertical sync 59.94 Hz

- จำนวน pixel จริงในแต่ละเส้นภาพ รวม 800 pixels
    - 8 pixels front porch
    - 96 pixels horizontal sync
    - 40 pixels back porch
    - 8 pixels ขอบภาพซ้าย
    - 640 pixels ภาพจริง
    - 8 pixels ขอบภาพขวา
- จำนวนเส้นภาพแนวตั้งจริง รวม 525 เส้น
    - 2 เส้น front porch
    - 2 เส้น vertical sync
    - 25 เส้น back porch
    - 8 เส้น ขอบภาพบน
    - 480 เส้น ภาพจริง
    - 8 เส้น ขอบภาพล่าง

Block Diagram



อธิบายโมดูล

การออกแบบ

เกี่ยวกับเกม

เป็นเกมที่ ให้ตัวละคร หลบศัตรูที่เกิดอย่างสุ่มและสามารถยิงปืนเพื่อฆ่าศัตรูได้ แต่เมื่อครบทุก ๆ 10 คะแนนศัตรูจะเกิดใหม่

ปืนสามารถกำจัดแมวได้ ปืนสามารถยิงได้5นัด แต่เมื่อกำจัดแมวไปแล้ว จะไม่สามารถยิงได้อีก และถ้าศัตรูชนตัวละครของเราจะทำให้ขึ้นหน้าจบเกม โดยการแสดงคะแนนที่ทำได้

การคิดคะแนน

จะนับตามเวลา โดยคะแนนจะเพิ่ม 1 คะแนนทุก ๆ ประมาณ 2.4 วิ สูงสุดที่ 99 คะแนน

การบังคับ

ใช้ปุ่ม 6 ปุ่ม

PB1-PB4 เป็นปุ่มบังคับทิศทาง

PB5 เป็นปุ่มยิงกระสุน

DIP SW1 เป็นปุ่มreset

ภาพรวม

เราสามารถแบ่งองค์ประกอบวงจรได้ 4 ส่วนหลัก คือ

1. ส่วนตัวละคร ทำหน้าที่ประมวลผลการเคลื่อนไหวของตัวละครหลัก
2. ส่วนลูกกระสุน ทำหน้าที่ประมวลผลการเลือกลูกกระสุนที่จะใช้ยิงและการเคลื่อนที่ของลูกกระสุนแต่ละลูก
3. ส่วนศัตรู ทำหน้าที่กำหนดการเกิดใหม่ ทิศทางและการเคลื่อนไหวของศัตรูแต่ละตัว
4. ส่วนกลาง ทำหน้าที่สร้างสัญญาณนาฬิกากลาง (สัญญาณกำหนดเฟรมภาพ, สัญญาณ sync, ตัวเลขสุ่ม) พิกัดภาพปัจจุบัน ประมวลผลการชน การนับคะแนน และการแสดงผลสู่จอภาพ

ส่วนตัวละคร

drawMainCharacter

ทำหน้าที่วาดภาพตัวละครจากตำแหน่งที่กำหนด โดยรับ input สัญญาณนาฬิกา ตำแหน่งพิกัดภาพบนหน้าจอที่กำลังวาด ซึ่งมาจาก hsync_generator และพิกัดของตัวละคร หาส่วนต่างระหว่างกันแล้วนำไปเปรียบเทียบในชุดเงื่อนไขว่าในพิกัดนั้นจะต้องแสดงสีใด ทั้งนี้ output ที่ได้เป็น 3-bits RGB code เพื่อนำไปใช้ในการเปรียบเทียบในโมดูลหลักก่อนนำไปแสดงผล

การสร้างภาพ ณ พิกัดใด ๆ ขององค์ประกอบอื่น (drawEnemy – วาดศัตรู, drawBullet – วาดลูกกระสุน, drawScore – วาดคะแนน) ล้วนใช้หลักการเดียวกัน

การสร้างชุดเงื่อนไขเพื่อใช้ในการเลือกว่าจะต้องแดสงสีใดถูกสร้างจากโปรแกรมแปลงรูปภาพเป็นชุดคำสั่งที่ทำขึ้นเอง รายละเอียดการสร้างอยู่ในปัญหาและวิธีการแก้ไข

ส่วนลูกกระสุน

bulletPosition

ทำหน้าที่ระบุทิศทางและตำแหน่งปัจจุบันของลูกกระสุน

ChangeBullet

ทำหน้าที่ตรวจสอบว่าลูกกระสุนใดว่าง และเรียกใช้ฯ เมื่อทำได้

drawBullet

ทำหน้าที่วาดภาพกระสุนของแต่ละลูกกระสุน

ส่วนศัตรู

enemyPosition

ทำหน้าที่สุ่มการเกิดของศัตรูและกำหนดทิศทางการเคลื่อนไหวและการกระทบชิ่งของศัตรูกับกำแพง

drawEnemy

ทำหน้าที่วาดภาพศัตรูบนจอ

ส่วนหลัก

random

ทำหน้าที่สร้างเลข (ลำลอง) สุ่มโดยอาศัย Linear Feedback Shift Register บวกกับส่วนหนึ่งของชุดตัวเลขสุ่ม (salt) และจำนวนครั้งของสัญญาณนาฬิกา ณ ขณะนั้น

hvsync_generator

ทำหน้าที่สร้างสัญญาณ sync ของ VGA และตัวเลขระบุพิกัดการวาดหน้าจอ ณ ขณะนั้น

DrawScore

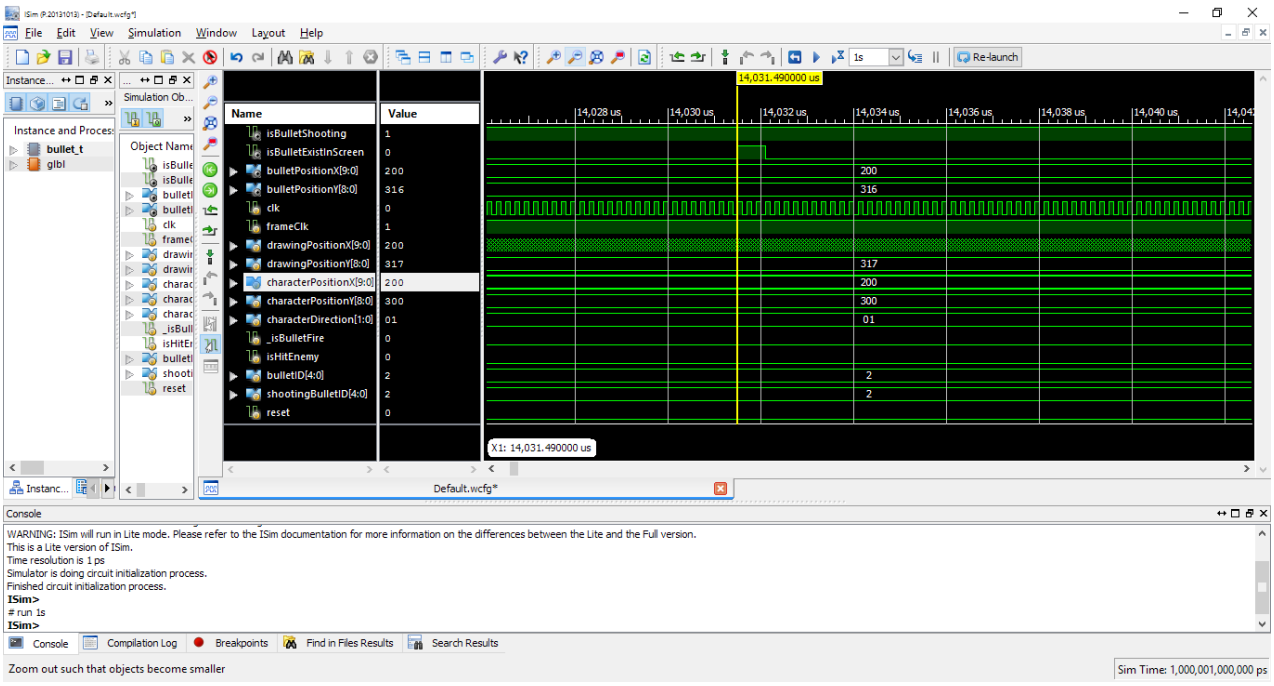
ทำหน้าที่ในการคำนวณคะแนนและวาดคะแนน ซึ่งจะรับค่า ตำแหน่งตัวละคร กระสุนปืน และ ศัตรู และคำนวณฟังก์ชันต่าง ๆ ของเกม ดังนี้

- การจบเกมเมื่อตำแหน่งศัตรูตรงกับตัวละคร(เมื่อชนกัน)
- การลบศัตรูเมื่อกระสุนชนศัตรู
- การจำกัดการยิงกระสุนให้ไม่สามารถยิงได้ เมื่อยิงโดนศัตรูไปแล้ว
- การคิดคะแนน ให้เพิ่มคะแนน ทุก2.4 วินาที
- การวาดคะแนนโดยใช้ *case* เพื่อเลือกคะแนนที่จะแสดงผลจากคะแนนที่คำนวณมา

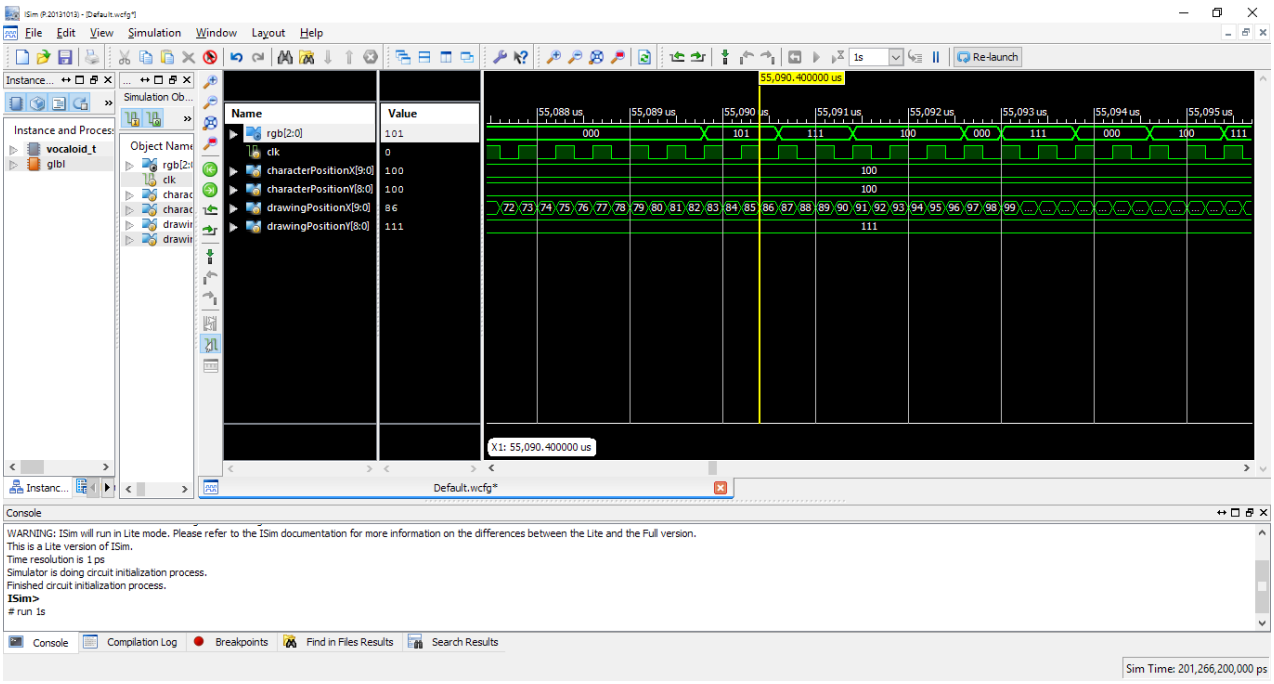
TopModule

ทำหน้าที่เรียกโมดูลย่อย ๆ ออกมาใช้งาน และส่งค่าต่าง ๆ ตามที่อธิบายไว้ในโมดูลต่าง ๆ และทำหน้าที่ หาตำแหน่งตัวละครเพื่อส่งไปที่โมดูลวาดตัวละคร

ผลที่ได้



รูปที่ 2 ตัวอย่าง Timing Diagram ของ bulletPosition ในจังหวะที่พิกัดวาดรูปตรงกับตำแหน่งปัจจุบันของกระสุน



รูปที่ 3 แสดง TIming diagram ของ drawMainCharacter ขณะที่ถึงจุดที่ตัวละครตั้งอยู่

รูปที่ 4 แสดงตัวอย่างหน้าจอขณะเล่น

รูปที่ 5 แสดงคะแนนหลังจากจบเกม

สรุปผล

เราสามารถประยุกต์ใช้ FPGA เพื่อสร้างเกมดังกล่าวในวัตถุประสงค์ได้โดยการสร้างโมดูลเพื่อควบคุมการแสดงผลและรับ input และประมวลผล

ปัญหาและวิธีการแก้ไข

ปัญหา แต่เดิมตั้งใจให้แต่ละส่วนกำหนดพิกัดลงบนตัวแปรแผนที่กลางแล้วจึงนำข้อมูลนั้นไปประมวลผล แต่บอร์ดมีหน่วยความจำชั่วคราวน้อย และ**ไม่ได้รับอนุญาตให้ใช้หน่วยความจำเสริม**

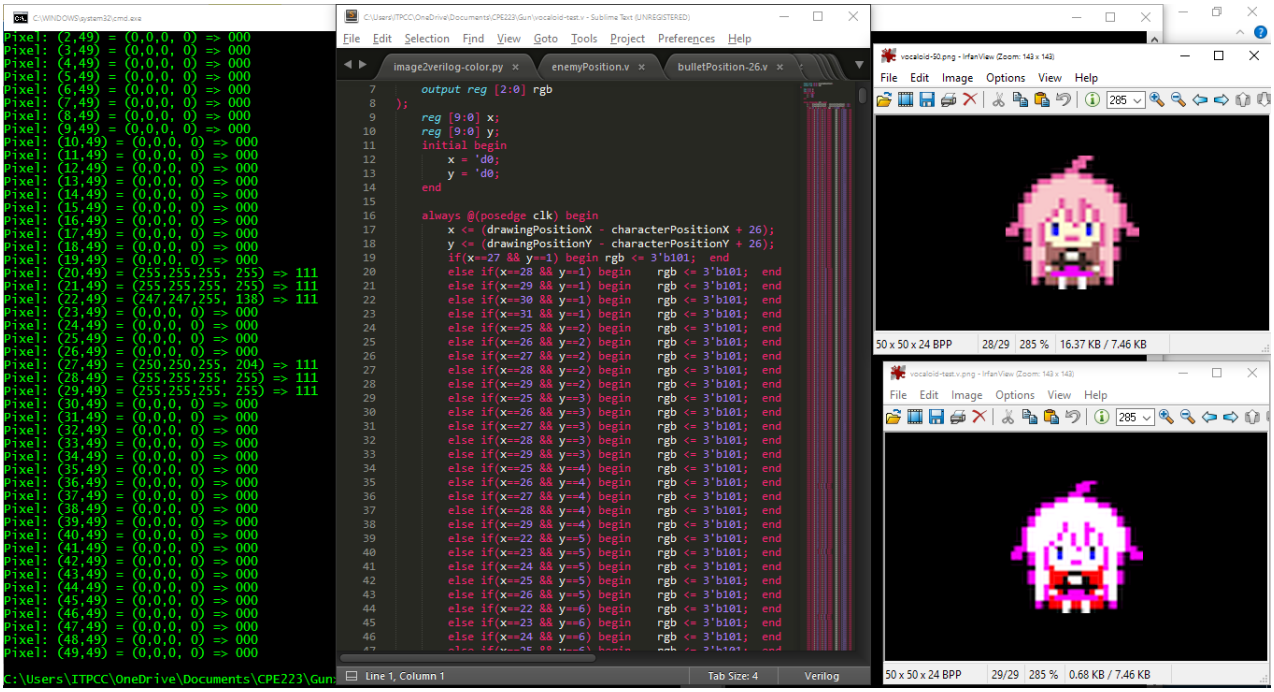
> **วิธีการแก้ปัญหา** ให้แต่ละส่วนวาดภาพออกมาโดยตรง (Output เป็นรหัสสี 3 bits RGB) แล้วประมวลผลการชนและการนับคะแนน ณ ขนะนั้นโดยทันที แม้จะทำให้มีชุดคำสั่งไม่พึงประสงค์ขึ้นมาบ้าง แต่สามารถบรรลุเป้าหมายได้ในที่สุด

ปัญหา แม้จะสามารถใส่ชุดคำสั่งได้ทั้งหมด แต่การประมวลผลเป็นชุดคำสั่งสำหรับ FPGA เริ่มช้าลงเรื่อย ๆ

> **วิธีการแก้ปัญหา** พยายามพัฒนา ลดชุดคำสั่ง หรือยุบรวมคำสั่งที่ใช้ร่วมกันได้ เพื่อลดคำสั่งให้ลดลง

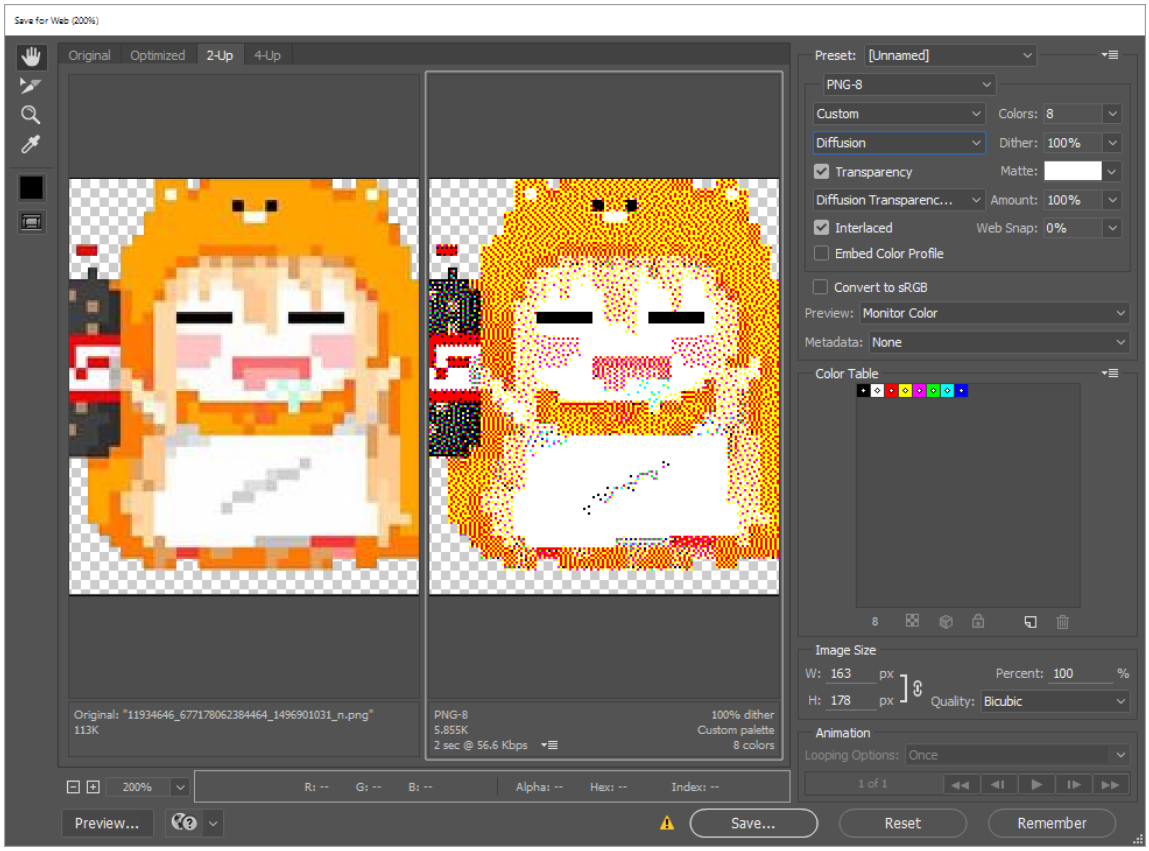
ปัญหา FPGA ไม่มีหน่วยความจำที่สามารถเข้าถึงได้ง่ายในการบรรจุภาพต้นแบบของตัวละคร ลูกกระสุน ศัตรู และคะแนน

> **การแก้ปัญหา** เปลี่ยนเป็นการสร้างชุดเงื่อนไขเพื่อดูพิกัดที่จะต้องวาดภาพ แล้วส่งรหัสสีผลลัพธ์โดยตรง
>
> ทั้งนี้ เพื่อความสะดวกและง่ายต่อการแปลงรูปภาพ จึงสร้างชุดคำสั่งภาษา Python ในการแปลงรูปภาพรูปแบบมาตรฐาน (ในที่นี้ใช้ภาพแบบ PNG เนื่องจากการบีบอัดไม่เสียคุณภาพและสามารถบันทึกจุดโปร่งใสได้) โดยการอ่านทุกพิกัดของภาพว่ามีความโปร่งใสน้อยกว่าครึ่ง (50) และแม่สี (แดง เขียว และน้ำเงิน) เกินค่ากลาง (126) ของแม่สีนั้นหรือไม่ หากใช่ให้นับบิตนั้นเป็น 1รวมเป็น 3-bits RGB พร้อมกับระบุพิกัดและรวมให้เป็นโมดูลในภาษา Verilog ซึ่งสามารถนำไปใช้ได้ทันที
>
> อย่างไรก็ตาม เนื่องจากการประมวลผลภาพที่สีต่างจากแม่สีหลักพอสมควร เช่นสีส้ม สีม่วง สีน้ำตาล อาจจะได้ภาพผลลัพธ์ไม่ตรงกับความต้องการ (ดูรูปที่ 6 ประกอบ) ในบางครั้งจึงนำภาพไปทำ Pattern Dithering ใน Adobe Photoshop หรือโปรแกรมแก้ไขรูปภาพอื่นที่มีคุณสมบัติใกล้เคียง ซึ่งเป็นการนำสีที่สามารถใช้ได้ในระบบฯ มาวางเป็นรูปแบบเฉพาะเพื่อให้ได้สีที่ใกล้เคียงกับสีที่ควรจะเป็น (ดูรูปที่ 7 ประกอบ)



รูปที่ 6 แสดงตัวอย่างการใช้ชุดคำสั่งแปลงภาพเป็นโมดูลในภาษา Verilog

(ซ้าย -- หน้าจอแสดงสถานะการแปลงภาพ, กลาง -- โมดูลผลลัพธ์, ขวาบน -- ภาพต้นฉบับ, ขวาล่าง -- ภาพตัวอย่างผลลัพธ์ที่ควรได้เมื่อนำไปใช้งานจริง) [© 2016 1st PLACE Co., Ltd.]



รูปที่ 7 แสดงการใช้ Pattern Dithering ในการจำลองสีอื่นๆ จากสีที่กำหนดให้ (ในที่นี้สร้างรูปผลลัพธ์ (รูปขวา) จาก 3 bits RGB) [©2015 サンクヘッド／集英社]

บรรณานุกรม

Alan Pemberton. 2014. Standard component video levels. *World Analogue Television Standards and Waveforms.* [ออนไลน์] Pembers' Ponderings, 28 December 2014. http://www.pembers.freeserve.co.uk/World-TV-Standards/index.html#Component.

APEX instrument CO., LTD. FPGA Discovery-III XC2S300 Board Manual V1.0. *APEX instrument CO., LTD.* [ออนไลน์] APEX instrument CO., LTD.[สืบค้นเมื่อ 28 March 2016.] https://lookaside.fbsbx.com/file/FPGA%20Discovery%20-%20III%20XC3S200%20%20Board_Manual_ENG_V1-2.pdf?token=AWwiSK_DmPPaNreAbVRz8jzwp0W5sr-d0olMbuOmLgA3d-Ubauzq4Qv85alfsiMmTHWXSbDIcZ-KqlWnLimJ2w5AB1nFrphvcRV2JgAuUsNPCeULsnNUvGfN26qbtG-otAI.

Chris Fletcher. 2008. Verilog: always @ Blocks. *Department of Electron Devices, Budapest University of Technology and Economics.* [ออนไลน์] UC Berkeley, 5 September 2008. [สืบค้นเมื่อ 26 March 2016.] http://www.eet.bme.hu/~nagyg/mikroelektronika/Always.pdf.

Eduardo Sanchez. 2007. A VGA display controller. *The Logic Systems Laboratory (LSL).* [ออนไลน์] Ecole Polytechnique Fédérale de Lausanne, 25 March 2007. http://lslwww.epfl.ch/pages/teaching/cours_lsl/ca_es/VGA.pdf.

Jean P. Nicolle. 2014. Pong Game. *FPGA4Fun.* [ออนไลน์] 12 April 2014. http://www.fpga4fun.com/PongGame.html.

Martin Hinner. 2007. "VGA industry standard" 640x480 pixel mode. *VGA (Video Graphics Array) Interface and video signal documents.* [ออนไลน์] January 2007. http://martin.hinner.info/vga/640x480_60.html.

—. 2007. VGA Signals. *VGA (Video Graphics Array) Interface and video signal documents.* [ออนไลน์] January 2007. http://martin.hinner.info/vga/vga.html.

ดร. ทรงยศ นาคอริยกุล. 2553. *การวิเคราะห์และออกแบบวงจรดิจิตอล.* กรุงเทพมหานคร : บริษัท แดเน็กซ์ อินเตอร์คอร์ปอเรชั่น จำกัด, 2553.